ARC ADDITIONS IN THE FLOW NETWORK

by

V. K. GUPTA

IMEP
1984
M
GUP
ARC

INDUSTRIAL AND MANAGEMENT ENGINEERING PROGRAMME
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
JULY, 1984

ARC ADDITIONS IN THE FLOW NETWORK

A thesis submitted
in Partial Fulfilment of the Requirements
for the degree of
MASTER OF TECHNOLOGY

by

V. K. GUPTA

to the

INDUSTRIAL AND MANAGEMENT ENGINEERING PROGRAMME
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
JULY, 1984

83746

IMEP-1984-M-GUP-ARC

POST GRADUATE OFFICE
Submitted on 25/7/84

CERTIFICATE

This is to certify that the work embodied in the thesis, "Arc Additions in the Flow Network," by Vijay Kumar Gupta has been carried out under my supervision and has not been submitted elsewhere for a degree.

July, 1984

(A. K. Mittal)
Assistant Professor
Industrial and Management En
Indian Institute of Technolo
Kanpur 208016

POST GRADUATE OFFICE
15/8/84

ACKNOWLEDGEMENTS

I express my deep sense of gratitude and feel deeply indebted to Dr. A. K. Mittal for his guidance and help at each stage of this work. His lively interest, devotion of his invaluable time and constructive criticism have been a constant source of inspiration and has made this work to attain its present stature. My association with him has gone a long way in building my attitudes on academic as well as non-academic issues.

I am very much thankful to my friends who have made my stay in the campus very pleasant.

I thank Mr. J.K. Misra for his immaculate typing and Mr. Buddhi Ram Kandiyal for neat cyclostyling.

V. K. Gupta

CONTENTS

ABSTRACT

In this thesis some special cases of the capacity expansion in the flow network are considered. Specifically the cases considered are: (i) From the given set of arcs which can be added to an existing network, select an arc such that it maximizes the increase in flow from source node to sink node. (ii) From the same set of arcs select arcs with minimum total capacity such that their addition to an existing network increases the flow from source node to sink node by a desired value.

For the first problem an implicit enumeration algorithm for which identifies the optimal arc is developed. For the second problem two heuristics and one exact method have been suggested. The exact method is an implicit enumeration scheme. The heuristics utilize the lower and upper bounds on capacity utilization of the arcs to be added.

The computational performance of the enumerative algorithm (for single are addition problem), implicit enumeration scheme and one heuristic method is tested on randomly generated networks.

CHAPTER I

INTRODUCTION

1.1 INTRODUCTION:

Physical distribution systems such as electricity or gas supply, road or rail networks, etc. can be easily modelled as networks of flow. Generally these systems are designed optimally with respect to the existing requirement. However with changes over time additions or contractions of these flow networks may be required.

After some time it may be desired to increase the flow in this network to a higher level. This can be done by either increasing the existing arc capacities or by introducing new arcs (or both). However, increase of this capacity will be at additional cost and as these costs are substantial, arcs with minimum cost would be the most desirable ones to be introduced in the network. This problem is very well known problem and considerable amount of literature is available on the capacity expansion of network problem.

However, there are cases of the problems where the capacity expansion may be done under restricted conditions. One simple case is when the costs of the arcs added are linear function of the capacity of the arcs. In such cases the problem of adding

arcs with minimum cost will be equivalent to the problem of adding arcs with total minimum capacity.

Similarly, we may be interested in finding the arc from the list of specified arcs whose addition will increase the flow to maximum. Such an arc can be viewed as most valuable arc.

In this thesis, we will concentrate on the above mentioned two problems. In the next few sections, we will review the literature in the related fields.

1.2 LITERATURE REVIEW:

There is considerable amount of literature available for the capacity expansion problem and related areas. For convenience we will review literature in the following areas.

(i) Maximum flow problem.
(ii) Minimum cardinality min-cut problem.
(iii) Network sensitivity analysis.
(iv) Capacity expansion problem.

1.21 Maximum Flow Problem:

Maximum flow problem can be briefly described as below: Given a network G(V, A) with vertex set V and arc set A and capacity c_{ij} on arc (i,j) find the maximum flow from a specified source node s to a specified sink node t such that flow is conserved at all the nodes except at source and sink and no arc has flow larger than its capacity. It is a well known problem and is a special case of linear program. Ford and Fulkerson s [3]

labelling algorithm was the first algorithm proposed for this problem, which essentially finds maximum flow through a series of augmenting paths. Algorithm is finite and gives a integer flow in the case when arc capacities are integer.

Dinic [2] has proposed an improvement over this algorithm. Dinic's algorithm involves constructing layered networks with respect to existing flow. These layered networks are the union of all shortest paths (in terms of number of links in path) between s and t, such that each path has flow augmenting capacity. A maximal flow in this network is obtained. The process is repeated till the maximum flow is obtained. The complexity of algorithm is $O(|V|^4)$, where $|V|$ denotes the cardinality of the vertex set V.

Karzanov [8] has further improved the method of finding maximal flow in the layered network, using concept of preflows. This algorithm has a overall complexity of $O(|V|^3)$. A similar but much simpler algorithm with complexity $O(|V|^3)$ is proposed by Kumar, Malhotra, Maheshwari [13]. A recent study [6] has compared the various algorithms for its computational efficiency and it concludes that Karzanov's and Dinic's algorithm perform better than other algorithms on average on randomly generated networks.

1.22 Network Sensitivity Analysis:

Problems of arc additions and arc deletions can be viewed as problems of sensitivity analysis on the network.

A most vital link in a flow network is the arc that reduces the flow by maximum amount when it is deleted from the network. Wolmer [16] has developed an algorithm to obtain the most vital link. He has also given the following necessary condition for a link to be a most vital link:

A necessary condition for an arc (a, b) to be most vital link is that for any maximum flow pattern in the network G(V, A) the flow in the arc (a, b) is at least as great as the flow over every arc in a minimum cut. An improvement on the algorithm developed by Wolmer has been made by Lubore, Ratliff, Siclia [12]. They identify the set of arcs satisfying the above necessary condition as candidate set. Then they have developed a labelling scheme similar to that of Ford and Fulkerson [3] labelling algorithm for obtaining maximum flow between given source and sink. This labelling scheme systematically searches for a flow path from node a to node b which does not include arc (a, b) such that flow may be diverted from arc (a, b) over this path where (a, b) is the arc from candidate set. If node b is labelled then a breakthrough occurs. The occurrence or nonoccurrance of breakthrough is used to ultimately determine the most vital link.

The n arcs in a network whose simultaneous removal from a connected single commodity flow network results in the greatest decrease in the throughput capacity of the remaining system between a specified pair of nodes (source and sink) are the n most vital arcs of the flow network.

Ratliff, Lubore, Scilia [15] have developed a efficient method to determine n most vital links of the network.

More precisely the problem of finding the n most vital link can be stated as that of finding a set of n arcs D_n^* in $G(N, A)$ such that min-cut separating s and t in $G(N, A-D_n^*)$ is less than or equal to min cut separating s and t in $G(N, A-D_n)$ for all sets $D_n \subseteq A$ with $|D_n| = n$.

A modified network procedure is developed first. Then the general algorithm which uses this modified network procedure is developed. If the modified network procedure fails to obtain the solution in one pass then General algorithm is used which uses modified network procedure several times to obtain the solution. It may be noted the problem of adding n arcs to increase max flow can be related with the removal of n arcs from the complete graph.

1.23 Min Cardinality Min Cut Problem:

The problem to be considered is finding minimum cuts in the given network with the additional property that the number of arcs in the cut $(X, \bar{X})$ or $(\bar{X}, X)$ or $(X, \bar{X}) \cup (\bar{X}, X)$ is minimized where $(X, \bar{X}) \cup (\bar{X}, X)$ is minimal cut of the given network.

Hamacher [5] has given a method for obtaining the min cardinality min cut. The theory of lexicographical network flows has been employed to develop the algorithm.

1.24 Network Capacity Expansion Problem:

Fulkerson [4] has considered the problem of allocating a budget of resources among the arcs of a network for the purpose of increasing its flow capacity relative to given sources and sinks. The assumption made is that the cost of increasing each arc capacity is linear. A labelling algorithm is described that permits rapid calculation of optimal allocation for all budgets. The above assumption permits direct formulation of the problem described above as a linear program. The algorithm produces solution to the problem, not only for a fixed budget, but for all budgets i.e., the problem is solved parametrically. The algorithm uses a variant of the labelling procedure previously developed to solve the maximum flow problems.

The long range planning model of transportation system to assist planners in assessing the impact of various levels of service in transportation network has been developed by Le-blanc[11] In particular rail road network distribution are considered, the emphasis being placed on the problem of determining optimal levels of service on each arc in the existing network.

The model is formulated so that improvement to the shipping arcs are included as decision variables. The concave transhipment problem is extended to the case where co-efficients of the shipping cost functions are treated as decision variables.

Thus the model determines optimum movement of freight between nodes and optimum improvements/degradations to the network to minimize shipping costs plus maintenance costs on the shipping arcs.

Christofides and Brooker [1] have considered the problem of increasing the capacity of existing network where new arcs of finite capacity can be added at certain cost. The problem considered is -Given an existing network, a list of arcs which could be added to the network, the arc costs and capacities and an available budget, the set of arcs are to be selected such that their addition will maximize the maximum flow from source s to sink t subject to the budgetary constraints.

The method describes an efficient tree search algorithm using bounds calculated by a dynamic programming procedure which are very effective in limiting the solution space explicitly searched. Computational results for a number of medium sized problems are described and computing times are reasonable.

Howard and Nemhauser [7] have considered the following problem: Given a sequence of predicted demands for N time periods, determine the optimal investment decision in each period to minimize a linear investment cost and a strictly convex cost capacity. The relationship between capacity and the investment decision is assumed to be linear, but time varying constraints on both the individual decisions and on the sum of decisions are considered. They have also derived an algorithm for solving this problem.

CHAPTER II

SINGLE ARC ADDITION IN A FLOW NETWORK

2.1 INTRODUCTION:

In this chapter we will consider the problem of adding a single arc in the flow network G(V, A) with a specified source vertex s and specified sink vertex t. We will like to select the arc to be added such that the increase in flow between s and t is maximized. Such a problem may be meaningful in case where the cost of addition is identical for all the arcs and the budget permits only addition of a single arc.

It is obvious that if such an arc is added between source and sink directly, it will result in the maximum increase in the flow. However, the choice of arcs to be added may be limited to set E and such an arc may not be there. Thus it will be worthwhile to explore the addition of arc in the case where no such s, t arc is permitted to be added, or the capacity of such an arc is small. In such case identification of such a link can be considered as most valuable link as its addition will increase the flow to maximum.

2.2 DEVELOPMENT OF THE PROBLEM:

We will consider a network G(V, A) where V is the node set and A is the arc set. Vertex s will denote the source and

vertex t the sink vertex. All arcs are undirected. Let c_{ij} denote the capacity permitted on the arc (i,j). A cut denoted by Y is a partition of the node set into two disjoint partitions Y and $\bar{Y}$ such that s ε Y and t ε $\bar{Y}$. Further capacity of cut Y is denoted as

$$C(Y, \bar{Y}) = \sum_{\substack{i\varepsilon Y \\ j\varepsilon \bar{Y}}} c_{ij}$$

The flow in this network is governed by the following theorem 2 given by Ford and Fulkerson [3].

Theorem 2.1 (Maxflow min cut theorem):

$$\text{Max flow } f = \min_{y} C(Y, \bar{Y})$$

An obvious corollary to above theorem is as follows.

Corollary 2.2:

An arc (i, j) will add to the max flow only iff it spans all the minimum cuts in the network, i.e. it adds to capacity of all minimum cuts.

It may be noted that if (s, t) arc is permitted to be added, then it will span all the cuts and hence will be the arc to increase the flow to the maximum extent.

We shall concentrate on the restricted version of the problem, where arc to be added is to be selected from the set E of arcs and the capacity of each arc is specified.

A simple enumerative scheme for solving this problem can be constructed by solving max flow problems after adding each arc individually and selecting the arc which increases the flow to a maximum value. However, the procedure will require $O(|E|\ |V|^3)$ effort. Enumeration can be substantially reduced by using corollary 2.2. As the selected arc has to span all the minimum cuts one can reduce the computation by obtaining a minimum cut Y, $\bar{Y}$ and selecting from the arc set E, set of admissible arcs as E' such that (i, j) ε E' if i ε Y, j ε $\bar{Y}$.

Let f_{ij} be the flow on the arc (i, j) corresponding to max flow f. Let P_i denote the node potential of node i i.e. amount of flow which can be sent from node s to node i ε Y or from node t to i i ε $\bar{Y}$, in the referant network. The referant network is constructed by adding two directed arcs corresponding to each undirected arc (i, j) with flow f_{ij}. The capacity of edge (i,j) will be $c_{ij} - f_{ij}$ and of ~~the~~ arc (j,i) will be $c_{ij} + f_{ij}$ in the referant network. Then the maximum flow which can be increased by adding an arc (m, n), is denoted by $p_{m,n} = \text{Min}\ (P_m, P_n, c_{mn})$. Also the flow will increase to maximum by adding the arc (p, q) which is max min (P_m, P_n, c_{mn}). In case E' contains arcs of the type $\underset{(m,n)\,\varepsilon\,E}{(i, t)}$ for i ε Y and (s, j) for j ε $\bar{Y}$ and the arcs are uncapacitated the maximum increase in flow will be on one (i, t) or (s, i) as the case may be.

Further reduction in computation can be obtained by identifying a minimum cardinality s cut and a minimum cardinality t cut. A cut Y is minimum cardinality s cut, if $|Y|$ is minimum among all minimum cuts. Similarly a cut Y is minimum cardinality t cut if $|\bar{Y}|$ is minimum among all such cuts. Let such cuts be denoted by $(S, \bar{S})$ and $(T, \bar{T})$ respectively.

Corollary 2.3:

An arc will increase the flow only if it is of type (i, j), $i \in S$, $j \in \bar{T}$. Thus E can be further reduced by selecting only those arcs which are of the above type.

Further in the following lemma, we show that min cardinality cuts $(S, \bar{S})$ and $(T, \bar{T})$ are unique.

Lemma 2.4:

Minimum cardinality cut $(S, \bar{S})$ (and therefore $(T, \bar{T})$) is unique.

Proof:

Proof follows from the following theorem, as stated in Ford and Fulkerson [3].

Theorem 2.5:

If $(X, \bar{X})$ and $(Y, \bar{Y})$ are two minimum cuts then $(X \cap Y)$, $\overline{X \cap Y}$ is also a minimum cut.

If $(S, \bar{S})$, the minimum cardinality cut is not unique, then there exists another minimum cardinality cut $(S', \bar{S}')$. Further

$|S \cap S'| < |S|$ and $(S \cap S', \overline{S \cap S'})$ is also a minimum cut, contradicting the fact that S is a minimum cardinality cut.

Next we describe a procedure to identify a minimum cardinality cut.

2.3 ALGORITHM FOR MINIMUM CARDINALITY CUT:

Procedure MINCARN:

Step 1: Solve a s, t max flow problem on the given network; obtain a min cut $(Y, \bar{Y})$.

Step 2: Increase the capacities of all the arcs to nc_{ij} where $n = |V|$ (number of nodes in the network).

Step 3: For all nodes $i \in Y$ add a arc (i, t) with $c_{it} = 1$ where c_{it} is the capacity of arc (i, t) or if there exists an arc (i, t), increase the capacity of arc (i, t) by 1.

Step 4: Solve a s, t max flow problem. The new min cut $(S, \bar{S})$, thus obtained will be min cut in the original network with the additional property that $|S|$ is minimum.

Proof: The capacity of the minimum cut in the modified network will be due to two types of arcs.

(i) The arcs already present in the original network.

(ii) The new arcs added (of (i, t) type) to modify the network.

Let the modified network be G'. Let the min cut in the original network G be K and in the modified network G' be K'. Let us assume that the min cut in the modified network is not a min cut in the original network.

As $|K'| \geq 1$, there will be at least one arc from node in K' to node t of capacity 1. Also since all arc capacities are integers, capacity of the cut K' in G',

$$C(K', G') \geq nC(K', G) + 1 \qquad (2.1)$$

where $C(K', G)$ is the capacity of cut in K' in G.

Also $C(K, G') \leq nC(K, G) + n - 1$ as $|K| \leq n-1$ (2.2)

where $C(K, G')$ is the capacity of cut K in graph G'. Now as K' is not a minimum cut in G,

$$C(K', G) \geq C(K, G) + 1 \qquad (2.3)$$

Hence from (2.1) and (2.3), $C(K', G') \geq nC(K', G) + 1$

$$\geq nC(K, G) + n + 1 \qquad (2.4)$$

Thus we have from (2.2) and (2.4),

$$C(K', G') \geq C(K, G')$$

which contradicts the fact K' is minimum cut in G'. Therefore K' will be minimum cut in G' only if it is a minimum cut in G.

Further, $C(K', G') = nC(K, G) + |K'|$ and as K' is minimum cut in G' it will be the cut with minimum $|K'|$ i.e. minimum cardinality cut in G.

2.4 UPPER AND LOWER BOUNDS ON ADDITIONAL FLOW BY ADDITION OF AN ARC:

As outlined in Section 2.2 the enumeration scheme for identifying the arc to be added to increase the maximum flow, will involve solving one max flow problem for each arc in E'. To reduce the complexity further we will explore the idea of associating a priority order for the addition of these arcs. For this we will identify procedures for computing lower and upper bound on the additional flow which can be obtained when a arc is added. We will identify lower and upper bound on the flow permitted by the nodes, for all nodes in minimum cardinality s cut, or minimum cardinality t cut and then compute the lower bounds on additional flow which can be obtained when specific arc is added.

2.41 Identification of Lower Bounds on Node Potential:

Given a maxflow pattern in the existing network corresponding to minimum cardinality cut $(S, \bar{S})$ the flow potential for each node in S can be obtained by solving maxflow problems between s and the node i in the network with residual capacities. However, this procedure will require solving a maxflow problem for each such node and will be time consuming. Lower bounds on each node potential can be obtained by identifying a good feasible flow between s and the node in the network with residual capacities. We will describe two procedures which obtain such flows.

2.42 Procedure LAYER:

In this procedure we shall first obtain a minimum cardinality s cut $(S, \bar{S})$ and a minimum cardinality t cut $(T, \bar{T})$ of the network G (V, A). Then a sequence of non-crossing cuts $S = S^k, S^{k-1}, \ldots S^o = s$ are obtained in a reduced network where all the nodes of $\bar{S}$ are condensed into a single sink node.

Further, let $V_k, V_{k-1}, V_{k-2}, \ldots, V_o$ denote the partitions of S such that $S^i = \bigcup_{j=0}^{i} V_j$. It may be noted $V_o = \{s\}$ and $V_i \cap V_j = \phi$.

Let $C(S^i)$ denote the capacity of cut $(S^i, \bar{S})$ in the original network. Then these non-crossing cuts are obtained such as to satisfy following conditions:

(1) $C(S^o) \geq C(S^1) \geq \ldots \geq C(S^k)$

(2) The node partitions are such that for any arc $(i, j) \in A$ if $i \neq s$, is in the node partition V_p, then j is in either node partition V_p or in V_{p+1} (or V_{p-1}). It implies that except from the source node all arcs will be either within same node partition or in adjacent node partition.

To obtain these cuts we iteratively follow following condensation procedure. If at the current step the current cut is $(S^i, \overline{S^i})$ in the network G (V, A) then condense the nodes of $\overline{S^i}$ in a single node t^i. Remove all the arcs from A which are in the cut $(S^i, \overline{S^i})$. For each node $p \in S^i$, if there is a arc in

cut $(S^i, \overline{S^i})$ add an arc (p, t^i) with capacity $c_{pt^i} = M$ (a large number $\geq \sum_{(i,j)\in A} c_{ij}$).

Solve a max flow problem between s and t^i. The minimum cut obtained corresponding to this maxflow will be S^{i-1}. Repeat the procedure till $S^o = s$.

In the above procedure following may be noted.

(i) $S^{i-1} \subseteq S^i$

(ii) If there is a node $p \in S^i$ which has an arc to a node $q \in S^j$, $j > i$ then node $p \notin S^{i-1}$ except when $i = s$. It follows from the fact that this node p will have an arc (p, t^i) with capacity M in the reduced network corresponding to $(S^i, \overline{S^i})$, and can never be capacitated, this implies that $p \in \overline{S^{i-1}}$, and hence $p \notin S^{i-1}$. However, if $p = s$ then s having unlimited capacity will belong to S^{i-1} ~~= s~~.

(iii) Partition $V_i = S^i - S^{i-1}$

(iv) $V_{i+1} \cap V_i = \phi$

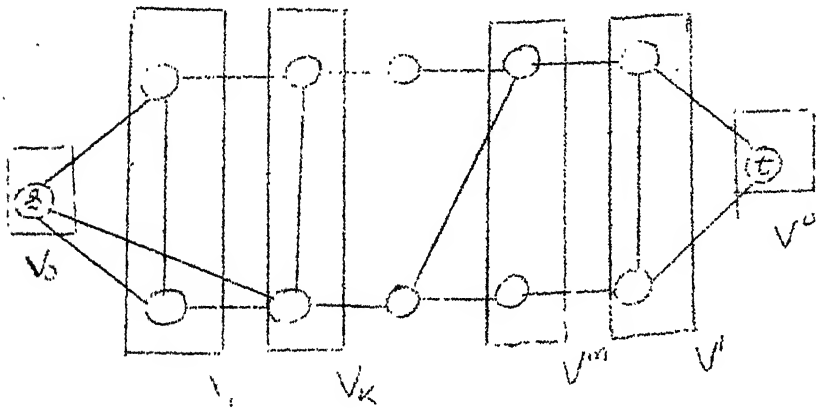
(v) From (ii) it follows that for any arc (p, q) if $p \in V_i$ then $q \in V_i$ or V_{i-1} or V_{i+1} except when $p = s$.

(vi) Further $C(S^{i-1}) \geq C(S^i)$ as $(S^{i-1}, \overline{S^{i-1}})$ is a cut in the condensed network in which minimum cut S^i was obtained.

Similarly corresponding to minimum cardinality t cut $(T, \overline{T})$ we shall obtain a sequence of noncrossing cuts $\overline{T^o}, \overline{T^1}, \ldots, \overline{T^m}$

such that $\overline{T^0} = \overline{T}, \ldots, \overline{T^m} = t$. These cuts will also have the properties similar to properties of V_i stated above. These cuts can be obtained by the procedure outlined above except that t will replace s in the procedure.

The resulting network will have the following structure



2.43 Algorithm LAYER:

Input: Network G (V, A) with capacities c_{ij} of arc (i, j) and with specified source node s and sink node t.

Step 0: Find the min cardinality s-min cut in G(V, A) using the algorithm MINCARN of Section 2.3. Let S be the partition set of the cut such that s ε S.

$k \leftarrow 0$

$X \leftarrow S$, $\bar{X} \leftarrow V - S$

Comment: X is the current cut partition containing node s.

Step 1: For all (i, j) s.t. $i \in X$, $j \in \bar{X}$,

$c_{ij} \leftarrow 0$

For all $i \in X$ s.t. there is an arc (i, j) s.t. $j \in \bar{X}$, add an arc (i, t),

$c_{it} \leftarrow M$ (Comment: M is large number).

Step 2: Solve maxflow between s, t and let the minimum cut obtained be $(S, \bar{S})$ then,

$\bar{V}_k \leftarrow X - S$

$X \leftarrow S$, $\bar{X} \leftarrow V - S$

IF $|X| = 1$ GO TO Step 3, else $k \leftarrow k+1$, GO TO Step 1.

Step 3: For $i = 0, k$, $V_i \leftarrow \bar{V}_{k-i}$.

Step 4: Output: $V_0, V_1, \ldots, V_k$.

Similarly the node partition of the minimum cardinality t cut can be obtained by interchanging s and t. STOP

2.44 Lower Bounds on Node Potential Using Procedure LAYER:

Once the layered network and the node partitions $V_k, V_{k-1}, \ldots, V_0, V^0, V^1, \ldots, V^m$ are obtained, the lower bounds on the potential of nodes $i \in S$ and $i \in \bar{T}$, can be computed in the following fashion:

For simplicity we shall renumber the nodes such that for all arcs if $i \varepsilon V_p$, and $j \varepsilon V_{p+1}$, then $i < j$, and arc is regarded as (i, j).

Compute for the each undirected arc (i, j), the residual capacity with respect to maximum flow as follows:

$r_{ij} = c_{ij} - f_{ij}$ if the flow on arc (i, j) is in direction (i to j).

$r_{ij} = c_{ij} + f_{ji}$ if the flow on the arc (i, j) is in direction (j to i).

Compute the lower bound ℓ_j on the node $j \varepsilon V_p$ as follows:

Let $a_j = \sum_{(i,j)\,|\,i \varepsilon V_{p-1}} r_{ij} + r_{sj}$

then,

$$\ell_j = a_j + \min_{i \varepsilon V_p} \{a_i, r_{ij}\}$$

a_j denotes the amount of flow which can be pushed to node j directly from the nodes in the previous layers. As the layer V_p is constructed by shrinking the nodes in $\overline{S_p}$, there exists feasible flow such that all edges between V_{p+1}, V_p will be saturated, and hence node j can receive at least additional flow a_j. Further from each node i in the same layer it can get flow which is minimum of a_i and r_{ij}. Thus ℓ_j are feasible additional flows from node s to node j.

2.45 Procedure REFERANT Network:

A simple procedure for constructing a layered network can be developed based on Dinics [2] method for max flow problem. In this procedure, first a maximum s, t flow is obtained. Next a directed network having two arcs corresponding to each arc in the original network is constructed. For convenience we shall label all undirected arcs as labelled in the direction of the flow. If f_{ij} is the flow on arc (i, j) from node i to node j and the capacity of this arc is c_{ij}, then the new network will have two arcs (i, j) and (j, i) withe capacity $(c_{ij} - f_{ij})$ and $(c_{ij} + f_{ij})$ respectively. If for any arc in this network the residual capacity is zero, the arc is removed from the network.

Construct the layers V_o, V_1, ..., V_m of the nodes as follows:

$$V_o = \{s\}$$

$$V_p = \{j \mid (i,j),\ i \in V_{p-1},\ j \notin \bigcup_{k=0}^{p-1} V_k \}$$

Let V_o, V_1, ..., V_m be the partitions of the min cardinality s-min cut, constructed by this procedure. A simple lower bound can be obtained by computing the arc disjoint path from s to node j and then adding the residual capacity of each arc disjoint path. The residual capacity of arc disjoint path is the residual capacity of minimum residual capacity arc in the path.

Thus $\ell_j = \sum_{p \in P} r_p$ where P is the set of arc disjoint path between s and j and r_p denotes the minimum residual capacity (augmentation capacity) of path P.

We next describe an algorithm to obtain these lower bounds.

2.46 Algorithm REFERANT Network:

Input: G(V, A) with c_{ij} as arc capacity ofarc (i, j) and specified source and sink. Compute maxflow f and flow f_{ij} on the arcs using Algorithm MINCARN. Let $(S, \bar{S})$ be the min cardinality s cut.

Step 1: For every arc (i, j) ϵ A

If $f_{ij} > 0$ then $\bar{c}_{ij} \leftarrow c_{ij} - f_{ij}$

$\bar{c}_{ji} \leftarrow f_{ij} + c_{ij}$

$V_o \leftarrow s$, $p \leftarrow 1$

$\bar{A} \leftarrow \phi$

Step 2: $V_p \leftarrow \phi$, $k \leftarrow 0$

Repeat for all (i, j) such that $i \in V_{p-1}$.

If $\bar{c}_{ij} > 0$ and $j \notin V_{p-1}$

$V_p \leftarrow V_p \cup \{j\}$

$k \leftarrow k+1$

$\bar{A} \leftarrow \bar{A} \cup \{i,j\}$

else delete arc (i, j).

If k =0, m$\leftarrow$p, GO TO Step 4.

Step 3: $p \leftarrow p+1$, GO TO Step 2.

Step 4: Output $V_0, V_1, \ldots, V_m$, S, $\bar{A}$. STOP.

Similarly algorithm is repeated to obtain partitions $V^k, V^{k-1}, \ldots, V^0$ of minimum cardinality t cut $\bar{T}$.

2.47 Procedure RLBOND:

Comment: This procedure computes the lower bounds on the node potential of nodes in S (or in $\bar{T}$). The lower bound on the node j is the sum of the residual capacity (flow augmentation capacity) of arc disjoint path from s to j.

INPUT: Network G(V, A), Capacity c_{ij} of arc (i, j).

Step A: Using procedures REFERANT Network obtain $V_0, V_1, \ldots, V_m$, S, $\bar{A}$, c_{ij}, $r_{ij} = \bar{c}_{ij}$

Step 0: $p \leftarrow 1$

$B_j \leftarrow 0$, for all $j \in V$

$F_{ij} \leftarrow 0$, for all $i, j \in \bar{A}$

Step 1: Find a node $j \in V_p$ such that $B_j = 0$.

$z \leftarrow M$ (Comment: large number)

GO TO Step 2.

IF no node $j \in V_p$ such that $B_j = 0$ is available,

GO TO Step 7.

Step 2: $q \leftarrow j$, $\ell \leftarrow 1$

Step 3: Find an arc (k, q) such that $k \varepsilon V_{p-\ell}$ and $F_{pq} = 0$

$z \leftarrow \min \{z, r_{kq}\}$

$F_{kq} \leftarrow 1$

GO TO Step 4.

IF no such arc is available, GO TO Step 6.

Step 4: $\ell \leftarrow \ell+1$, $q \leftarrow k$

IF $(p - \ell < 0)$, GO TO Step 5, else GO TO Step 3.

Step 5: $j \leftarrow \ell_{j+z}$, GO TO Step 2.

Step 6: $F_{kq} \leftarrow 0$, for all $(k, q) \varepsilon \bar{A}$

$B_j \leftarrow 1$

GO TO Step 1.

Step 7: $p \leftarrow p+1$

IF $p \leq m$, GO TO Step 1, else GO TO Step 8.

Comment: m is the number of layers in the network G (V, $\bar{A}$).

Step 8: Lower bounds on nodes have been computed output ℓ_j. STOP.

2.48 Upperbound on the Node Potential:

Simple upperbound for each node $i \varepsilon S$ can be computed by noticing that the maximum amount of flow a node can receive from the source is equal to sum of the capacity of arcs incident on this node which have one end in min cut partition S less the sum of the capacity of arcs which have one end in min cut partition $\bar{S}$ and are incident on this node i.e.

$$u_i = \sum_{(i,j)\varepsilon B_i \cap S} c_{ij} - \sum_{(i,j)\varepsilon B_i \cap \bar{S}} c_{ij} \quad .$$

where B_i is the set of arcs which are incident on node i.

2.49 Upper and Lower Bounds on the Arcs:

Let E' constitute the feasible set of arcs from E which can be added to network G (V, A).

Arc potential of an arc of E' is the amount of flow it can increase in the network.

We obtain the lower and upper bound on arc potential for every arc of E' in the following fashion.

Lower bound on arc potential of arc (i, j) ε E' is given by the following equation.

$$p_{ij} = \min \{ \ell_i, \ell_j, c_{ij} \}$$

where ℓ_j is the lower bound on node potential of node j and c_{ij} is the capacity of arc (i, j).

Upper bound on arc potential of arc (i, j) ε E' is given by the following equation:

$$V_{ij} = \text{Min } \{ u_i, u_j, c_{ij} \}$$

where u_j is the upper bound on node potential of node j.

2.50 Procedure for Determining the Arc Causing Maximum Increase in Flow:

In this section we give a method for selecting an arc from the given set of arcs E', such that when it is added to the network $G(V, A)$ the increase in flow from s to t is maximum.

In this enumerative scheme arcs in E', will be arranged by a given priority rule in a set Q and will be selected from Q one by one. The four priority orders we suggest are as follows:

(a) $Q = E'$ i.e. arcs in Q are arranged in the same order as that in E'.

(b) Q is the set of all arcs ε E' which are ordered in the non increasing order of the lower bounds on arc potential.

(c) Q is the set of all arcs ε E' which are ordered in the non increasing order of the upper bounds on arc potential.

(d) Q is the set of all arcs in E' which are ordered in the non increasing order of the capacities of arcs.

At a general step an arc from Q is selected and its upper bound an arc potential is tested with the incumbent solution. Depending upon whether the upper bound is greater than the incumbent solution or not the arc is added to the network or pruned from further consideration. The incumbent solution is duly updated whenever a better solution is obtained. At termination all arcs of Q have been examined and the incumbent gives the optimal solution.

However in case Q is constructed by priority order (c), then the first time a incumbent greater than the upper bound of arc under consideration has been obtained gives the optimal solution.

The procedure outlined in this section can be implemented by the following algorithm.

2.51 Algorithm ADD:

INPUT: Network G (V, A), arc capacity c_{ij} specific source s and sink t. Priority list Q, upper bound on the arc potential of arcs of Q, Maxflow f.

Step 0: $k \leftarrow 1$, $f^* \leftarrow 0$

Comment: f^* denotes the current (maximum) additional flow obtained.

Step 1: Select the k-th arc (i,j) from Q. IF $f^* \geq v_{ij}$, GO TO Step 2, else
IF $f^* < v_{ij}$ then add (i,j) to G(V, A).
Solve maxflow between s, t and let the flow be f_k.

$g \leftarrow f_k - f$

IF $g > f^*$ then $f^* \leftarrow g$ and $p \leftarrow i$, $q \leftarrow j$, GO TO Step 2.

Comment: If priority set Q is according to the non-increasing order of upper bounds on arc potentials then at the first violation of condition $v_{ij} > f^*$, GO TO Step 3.

Step 2: $k \leftarrow k+1$

IF $k > |Q|$, GO TO Step 3,

else GO TO Step 1.

Step 3: (p, q) is the desired optimal arc and the optimal increase in flow in the network $G(V, A)$ is f^* . STOP.

2.52 Computational Performance:

The algorithm described above has been coded in FORTRAN-10 for DEC System 1090. A listing of the program is given in the Appendix.

The program is tested on randomly generated problems. Test networks were generated as follows: The size parameters i.e. number of nodes, number of arcs and length and breadth of the network, were fixed. Then a skeleton network consisting of arc disjoint path was constructed. The end points of the remaining arcs were generated randomly using uniform random number.

The capacities of arcs J was generated using a uniform random number with range as $20|H(J) - T(J)|$, $160|H(J) - T(J)|$, where H(J) and T(J) are the end points of the arc J. This permits the generation of the ~~one~~ arc capacity from uniform distribution with different ranges.

The problems of size ranging from 30 nodes to 200 nodes with density from .1 to .8 were generated.

In Table 2.1 a comparison of problems of 30 nodes to 200 nodes with different arc set combinations is given. Alongwith the comparison of execution time for 3 priority rules the number of maxflow computations required for each priority rule is also given. For these problems the average execution time required by P_1 is about 32 percent less than that required by P_2.

As such we suggest that priority rule P_1 may be used.

CHAPTER III

MULTIPLE ARC ADDITION IN A FLOW NETWORK

3.1 INTRODUCTION

In this chapter we will consider the problem of adding multiple arcs in the flow network with a specified source vertex s and sink vertex t.

We will like to select the arcs with minimum total capacity, whose addition will permit the flow to increase to the desired level. This problem is meaningful in the situations where the total cost of arc addition is a linear function of the total capacity of arcs. We shall further restrict ourselves to the selection of the arcs from the specified set E' which spans the min cardinality minimum cuts. Essentially it is the extension of the previous problems in which only one arc was permitted to be selected.

We will consider a network G(V, A) where V is the node set and A is the arc set, with vertex s as the source and vertex t as the sink. All arcs are undirected. Let c_{ij} denote the capacity of an arc (i, j). E' denotes the set of arcs which span the min cardinality minimum cuts. E' can be obtained as described in Section 2.3.

A subset of E' will be termed as feasible subset if the total sum of the capacity of the arcs in the subset is higher than the specified flow.

A simple enumerative scheme for solving this problem can be constructed by solving max flow problem after adding each feasibl subset to the network. The optimal subset is a subset which increase the flow to the desired value and for which total capacities of the arcs is minimum. The computational effort required for such a scheme will be very high as there can be large number of feasible subsets of arcs in E'.

We will describe an exact algorithm to solve this problem in Section 3.3. Exact algorithm is an implicit enumeration scheme and will require a procedure to obtain all subsets of a given set of elements. In Section 3.2 we describe one such procedure NEXKSB. This procedure is given in [14].
In Section 3.4 we will describe heuristic methods to solve this problem and will discuss computational performance in Section 3.5.

3.2 PROCEDURE NEXKSB [14]:

Let us consider the combination of n things taken k at a time. There are $\binom{n}{k}$ k subsets of an n-set. This procedure will generate them all in alphabetical order. In the lexicographical sequence, we obtain the successors of a given k-subset for n-set. For example let the n-set be Q = {1,2,3,4,5}. The 3-subsets of Q in lexicographical sequence are given below:

83746

{1,2,3}, {1,2,4}, {1,2,5}, {1,3,4}, {1,3,5}, {1,4,5}, {2,3,4}, {2,3,5}, {2,4,5} and {3,4,5}.

In the lexicographical sequence, we obtain the successor of a given k-subset $\{a_1, \ldots, a_k\}$ as follows. Search for the smallest h such that $a_{k+1-h} < n+1-h$, then increase a_{k+1-h} by 1, and set $a_j \leftarrow\!\!- a_{j+1}$ $(j = k+2-h, k)$. It is interesting to note that the index h can be found without searching. Indeed, at each transition from a set to its successors, h increases by 1 unless $a_{k+1-h} < n-h$, in which case h is reset to 1 on the next transition.

Algorithm NEXKSB [14]:

(1) $m \leftarrow\!\!- 0$; $h \leftarrow\!\!- k$; GO TO 4 {first entry}

(2) IF $m \geq n-h$, GO TO 3; $h \leftarrow\!\!- 0$ {later entries}

(3) $h \leftarrow\!\!- h+1$; $m \leftarrow\!\!- a_{k+1-h}$

(4) For j = 1, h: $\{a_{k+j-h} \leftarrow\!\!- m+j\}$
If $a_1 = n-k+1$ GO TO 5

(5) EXIT.

3.3 EXACT METHOD FOR SELECTING ARCS FROM E':

Next we shall describe an implicit enumeration scheme for selecting the minimum capacity arcs.

Procedure IMSEARCH:

Initialize: Let the arcs of E' be ordered as
$(a_1, a_2, \ldots, a_m)$, where
$c(a_1) \leq c(a_2) \leq \ldots \leq c(a_m)$, ~~where~~

where $c(a_i)$ is the capacity of arc a_i.

For convenience we shall denote $c(a_i)$ by c_i and the arc by a_i in this order by i. Further let g denote the increase in the flow desired.

Step 1: Select, first k arcs such that when these arcs are added to the network, flow is increased by atleast amount g. Let the capacity of these arcs be the current upper bound

$$UB = \sum_{i=1}^{k} c_i$$

Step 2: For each subset P of the arcs$(a_1, \ldots, a_{k-1})$ examine if

$C(P) + c_k \geq g$ and

$C(P) + c_k < UB$.

If yes solve a max flow problem and if the increase in flow $_P \geq g$ set UB = C(P). If all the subsets are enumerated go to Step 3 else repeat for next subset.

Comment: In the algorithm these subsets are generated such that first subsets of cardinality j = 1 are generated then subset of cardinality j + 1 and so on are generated. These subsets are generated using the procedure NEXKSB given in Section 3.2.

Step 3: $k \leftarrow k+1$

If $k > m$, go to Step 4 else select k-th edge from the list, add it to the list of selected edges and go to Step 2.

Step 4: Stop. Output, list of selected subset, capacity, flow.

In the next Section, we give a simple algorithm to execute this procedure.

3.31 Algorithm IMSEARCH:

INPUT: $G(V,A)$ with c_{ij} as capacity of arc (i,j), E', g (required flow increase).

Step 0: Solve Maxflow (A, f)

Comment: this procedure solves a maxflow problem between s and t in a network with arc in A and gives the maxflow as f.

$f^* \leftarrow f$.

Step 1: Arrange the arcs in E' in the nondecreasing order of the magnitude of their capacities. Let it be $(a_1, a_2, \ldots, a_n)$.

Let, $E' = (a_1, a_2, \ldots, a_n)$

$R \leftarrow \phi$, $k \leftarrow 1$, $M \leftarrow \phi$, $J \leftarrow 1$, $C(R) \leftarrow 0$.

Step 2: Select arc a_k from E'.

$R \leftarrow R \cup \{a_k\}$

$C(R) \leftarrow C(R) + c_k$

Step 3: If $C(R) \geq g$ GOTO Step 4, Else $k \leftarrow k+1$; GO TO Step 2.

Comment: $C(R)$ is the total capacity of all arcs of set R.

Step 4: $\bar{A} \leftarrow A \cup R$

Solve Maxflow $(\bar{A}, f)$

$e \leftarrow f - f^*$

If $e \geq g$, $NA \leftarrow k$, $U \leftarrow C(R)$

GO TO Step 5A

else $k \leftarrow k+1$, GO TO Step 2.

Step 5A: Repeat for all $i = 1, k-1$

$M \leftarrow M \cup \{a_i\}$

Step 5B: $L \leftarrow a_k$, $\bar{A} \leftarrow A \cup a_L$

$M \leftarrow M \cup a_{k-1}$

If $c_L \geq g$ and $c_L < U$

Comment: c_L is the capacity of L-th arc in E'.

Solve Maxflow $(\bar{A}, f)$,

$e \leftarrow f - f^*$

IF $e \geq g$ GO TO Step 9.

Else GO TO Step 6.

Step 6: Using Algorithm NEXKSB get next J-subset of M. Let it be X. If there is no subset of size J, GO TO Step 7.

If $c_L + C(X) < U$ and $\geq g$

then $\bar{A} \leftarrow \bar{A} \cup X$

Solve Maxflow $(\bar{A}, f)$

$e \leftarrow f - f^*$

IF $e \geq g$, $U \leftarrow c_L + C(X)$, $\bar{A} \leftarrow \bar{A} - X$

GO TO Step 6.

Step 7: $J \leftarrow J+1$

If $J \leq NA$ GO TO Step 6, else GO TO Step 8.

Step 8: If $|M| < n$, $k \leftarrow k+1$, $J \leftarrow 1$

GO TO Step 5B, else GO TO Step 9

Step 9: Output: Current incumbent, U, $\bar{A}$, STOP.

3.4 HEURISTIC METHODS:

In this section we will suggest the heuristics for solving the problem given in Section 3.1. The first heuristic LBOND, essentially orders the arcs in order of their utilization, where utilization is the fraction of the capacity which is likely to be utilized when the arc is added to the network. The second heuristic UBOND is simpler in execution but may not give as good solutions as LBOND.

Heuristic LBOND:

Initialize: Let the network by $G(V,A)$. Required flow increase g, Maxflow in this network f^*, E' the set of the arcs which can be added.

Step 1: Compute lower bound p_{ij} on arcs $(i,j) \in E'$ using algorithm of section 2.49.

Let $F = \{(i,j) | p_{ij} = c_{ij}\}$

order elements in F in nonincreasing order of c_{ij}.

$R \leftarrow E' - F$

For $(i,j) \in R$, compute $d_{ij} \leftarrow \frac{c_{ij} - p_{ij}}{c_{ij}}$.

Order arcs in R according to nondecreasing order of d_{ij}.

Let g' be the remaining flow increase required and $\bar{A}$ be the arc set of the current network.

$g' \leftarrow g$

$\bar{A} \leftarrow A$

Step 2: If $F = \phi$, GO TO Step 3 else select an arc in F whose capacity is nearest to g'. Add this arc to current network arc set $\bar{A}$.

Solve Maxflow $(\bar{A}, f)$. Let $e \leftarrow f - f^*$. If $e \geq g$ GO TO Step 4, else remove the selected arcs from F, repeat Step 2.

Step 3: Select the first arc in R, add it to $\bar{A}$, solve Maxflow $(\bar{A}, f)$. $e \leftarrow f - f^*$. If $e \geq g$, GO TO Step 4, else remove the selected arc from R, repeat Step 3 till R is empty.

Step 4: STOP. OUTPUT, $\bar{A}$, e

3.41 Algorithm LBOND:

In this section we describe an algorithm for the heuristic in Section 3.4.

INPUT: G (V, A), c_{ij}, E', g

Solve Maxflow (A, f)

$g' \leftarrow g$

Step 1: Obtain lower bounds on mode potential on the network G(V, A) by method of Section 2.44.

$\ell \leftarrow 1$

Step 2: Repeat for all $(i,j) \in E'$

$p_{ij} \leftarrow \text{Min } \{l_i, l_j\}$

$F \leftarrow \phi$

$R \leftarrow \phi$

Step 3: For all (i, j) such that $(i,j) \in E$,

If $p_{ij} \geq c_{ij}$,

then $F \leftarrow F \cup \{(i, j)\}$.

Arrange all arcs in F in the nonincreasing order of their lower bounds on arc potentials.

Step 4: For all $(i, j) \in E'$ such that $p_{ij} < c_{ij}$.

$$d_{ij} \leftarrow \frac{c_{ij} - p_{ij}}{c_{ij}}$$

$R \leftarrow R \cup \{(i, j)\}$

Arrange arcs in R in the non-decreasing order of capacity unutilization factor d_{ij}.

Step 5: If $F = \phi$, GO TO Step 7, else select the first arc (i, j) in F. If $p_{ij} \geq g'$ GO TO Step 6, else add (i, j) to $\bar{A}$. GO TO Step 8.

Step 6: $\ell \leftarrow \ell + 1$ $\ell \leftarrow \ell - 1$

If $\ell > |F|$, GO TO Step 6B, else select the ℓ-th arc, (i,j)

If $p_{ij} \leq g'$, repeat Step 6, else $\ell \leftarrow \ell - 1$.

Step 6b: Select ℓ-th arc. Add it to $\bar{A}$. GO TO Step 8.

Step 7: Select the first arc from R and add it to $\bar{A}$.

Step 8: Solve Maxflow $(\bar{A}, f)$

$e \leftarrow f - f^*$

If $e > g$, GO TO Step 9, else remove (i,j) from E',

$g' \leftarrow g - e$, $\ell \leftarrow 1$

GO TO Step 3.

Step 9: Output $(\bar{A})$. STOP.

Heuristic UBOND:

Initialize: Let E' be the set of arcs to be added, G (V, A) the current network, g as the required flow increase.

$\bar{A} \leftarrow A \qquad \bar{E} \leftarrow E', \; g' \leftarrow g$

Step 1: Compute upper bound on arcs of $\bar{E}$, and order them in order of nonincreasing upper bound. Let,

$$\bar{E} = \{a_1, a_2, \ldots, a_m\}.$$

Step 2: Select first k arcs from $\bar{E}$ such that sum of their upper bound is greater than the required flow increase. Add these arcs to A. Let the new arc set be $\bar{A}$. Solve Maxflow $(f, \bar{A})$. Let e be the increase in the flow, then if $e \geq g'$, GO TO Step 3, else remove these arcs from $\bar{E}$.

Let $\bar{E} = \bar{E} - \bigcup_{i=1}^{k} a_k$

$g' \leftarrow g - e$

GO TO Step 1.

Step 3: From $\bar{A}$ remove all those arcs of E' on which there is no flow. OUTPUT $\bar{A}$. STOP.

3.5 COMPUTATIONAL PERFORMANCE:

The exact algorithm and the heuristic LBOND described above has been coded in FORTRAN-10 for DEC System 1090.

The program is tested on randomly generated problems. Test networks were generated as follows: The size parameters i.e. number of nodes, number of arcs and length and breadth of the network were fixed. Then a skeleton network consisting of all disjoint path was constructed. The end points of the remaining arcs were generated randomly using uniform random number. The capacities of arc J was generated using a uniform random number with range from 20 to 200.

The problem of size ranging from 30 nodes to 150 nodes with density from 0.1 to 0.8 were generated. The cardinality of E was varied from very small values ($|E'| = 5$) to large values ($|E'| = 80$) and for each problem at least two values of increase in flow were specified.

In Table 3.1 a computational performance of these problems with different arc set combination is given. The heuristic LBOND is fast compared to the exact method when the cardinality of E' is large (> 20). However when cardinality of E' is small the two methods perform indiscriminately.

Also 94 percent of the solutions obtained by LBOND were within 25 percent of theoptimal value, and 62.5 percent of the solution obtained by LBOND were within 10 percentof the optimal value.

Table 3.1: Computational performance of the exact and heuristic methods.

S.No.	No.of nodes	No.of Arcs	No.of Problems Solved	Average $\|E'\|$	Computational Performance+						
					Heuristic LBOND				Exact Method		
					C_H	S_1	S_2	T_H	C_E	T_E	$K = T_E/T_H$
1	20	140	8	6	47	5	3	1.53	42.33	1.42	0.93
2	30	150	4	9.5	431.8	2	2	3.35	392.5	3.78	1.13
3	30	200	6	13	1044	4	2	3.85	944	5.50	1.43
4	30	250	3	14	337	1	1	5.39	308	11.17	2.07
5	30	300	2	28	1864	-	2	6.85	1478.5	19.91	2.90
6	50	300	4	11	413.5	3	1	4.14	403.25	5.78	1.40
7	50	400	4	22	408	3	-	4.83	405.66	7.25	1.50
8	50	600	5	26	267.3	4	-	9.57	256	16.16	1.69
9	50	800	2	19	260	2	-	9.58	256	15.6	1.63
10	100	600	4	69	2311	1	3	12.35	1944	71.75	5.80
11	100	800	2	4	329	2	-	12.37	329	11.3	0.91
12	100	800	2	73	2238	-	-	13.02	++	++	++
13	100	1000	2	4	423	1	1	10.56	393	10.60	1
14	150	1000	2	6.5	314	2	0	17.8	314	14.68	0.83
15	150	1200	2	47	549	-	-	33.41	++	++	++

+ C_H C_E : Average capacity of arcs added by heuristic and exact methods respectively.

T_H, T_E : Average CPU time taken by heuristic and exact methods respectively.

S_1 : No. of times heuristic solution was greater than the optimal solution by less than 10 percent of the optimal solution.

S_2 : No. of times the heuristic solution was greater than the optimal solution by less than 25 percent of the optimal solution.

++ : The execution did not stop even after 100 records of CPU time.

CHAPTER IV

CONCLUSIONS AND RECOMMENDATIONS

In this thesis we have considered the problem of expanding capacity of a flow network by adding arcs from a set of given arcs. We have considered two specific problems. In first problem, a single arc which increases the flow to maximum extend is required to be identified. In the second problem, we ~~will~~ select a subset of arcs from a given set of arcs such that the addition of these arcs can increase the flow capacity to a desired level and the sum of arc capacity is minimum among all such subsets.

An implicit enumerative method has been developed for selecting single arc optimally to maximize the flow increase in an existing network. The computation of lower bounds on arc flows by the method of condensing the network gives useful information which can be utilized efficiently to select the order in which arcs are to be added in the enumerative scheme. Method can be further applied by obtaining better upper bound on arc flows which will help in reducing the number of maxflow problems to be solved to obtain the best arc.

For the second problem, we have developed a implicit enumeration scheme and two heuristics. The implicit enumeration scheme for the multi arc addition problem gives efficient results in terms of computation time. Problem upto 150 nodes can

be solved using this algorithm in less than 30 seconds.

Since the problem considered in this thesis is a specific problem, a more general problem where arcs can be selected from any given set of arcs can be considered for further work. It may be noticed that the heuristic LBOND will work for any given set of arcs to be added, provided E' is replaced by the given set. For the exact method, however a new algorithm will have to be developed.

REFERENCES

1. Christofides, N., and Brooker, P., Optimal expansion of an existing network, Mathematical Programming, Vol. 6, No. 2, April 1974.

2. Dinic, E.A., Algorithm for solution of a maximum flow in a network by the method of preflows, Soviet Math. Dokl., Vol. 15, 1974.

3. Ford, L.R. and Fulkerson, D.R., Flows in Networks, Princeton University Press, 1962.

4. Fulkerson, D.R., Increasing the capacity of a network: The parametric budget problem, Management Science 5 (1959).

5. Hamacher, H., Determining Minimal cuts with a minimal number of arcs, Networks, Vol. 12, No. 4, 1982.

6. Hiroshi, I, On the practical efficiency of various maximum flow algorithms, Journal of the Operations Research, Society of Japan, Vol. 26, No. 1, March 1983.

7. Howard, G.T. and Nemhauser, G.L., Optimal Capacity Expansion, NRLQ, Vol. 15, No. 4, 1968.

8. Karzanov, A.V., Determining the maximal flow in a network by method of preflows, Soviet Math. Dokl., 15, 1974.

9. Kaltenbach, J.C., Peschen, J. and Geheig, E.H., A mathematical optimization technique for the expansion of electric power transmission system, Transactions of the Institute of Electrical and Electronic Engineers, 89, 1972.

10. Lawler, E.L., Combinatorial optimization networks and Matroids, Holt, Rinehart and Winston, 1976.

11. Le Blanc, L.J., Global solution for nonconvex, nonconcave Rail Network Model, Management Science, 23, 1976.

12. Lubore, S.H., Ratliff, H.D. and Scilia, G.T., Determining the most vital link in a flow network, NRLQ, Vol. 18, No. 4, 1971.

13. Malhotra, V.M., Kumar, M.P., and Maheshwari, S.N., An $O(N^3)$ Algorithm for finding maximum flows in networks, I.P.L., 1978.

14. Nijenhuis, A., and Wilf, H.S., Combinatorial Algorithms for Computers and Calculators, Academic Press, New York, 2nd Ed. 1978.

15. Ratliff, H.D., Scilia, G.T. and Lubore, S.H., Finding the n most vital links in flow networks, Management Science, Vol. 21, Jan. 1975.

16. Wolmer, R.D., Some methods for determining the most vital link in a Railway network, RAND Memorandum, RM-3321-ISA, 1963.

83746

IMEP-1984-M-GUP-ARC